

Raymond Gee  
John-Joshua Gutierrez  
Pyae Naing  
Dang Le  
Onubulachi Wami  
JunMin Li  
Nikhil Mohan  
Professor Parra  
CSC 667  
18 December 2019

### Final Project “Gator Dater” Write Up

For the final project we divided everyone into either a front-end or back-end role. We initially started with 6 people, so we had 3 people for the front-end and 3 for the back-end. The first thing we did was create a proposal and a UI Mockup, to get a general idea of how we wanted the application to look and operate. We decided that we would use Express for the back end for get and post endpoints, MongoDB for storing user information (on the cloud so that we are working on the same set of data, which hopefully results in the same results when testing), Amazon Web Service Elastic Compute Cloud (AWS EC2) to serve our application, Websocket for keeping track of clients, GridFs for storing files, React and Redux for building UI, Microserver architecture for the back-end to run multiple instances for some of the services, Gateway for proxy (so all front-end requests hits Gateway first for authorization), Redis for caching user login info (so they do not have to relog each time). We were not able to implement Docker and Kafka (for now).

To start, we used React Bootstrap to setup the UI, but slowly converted to Material-UI. This was because we had 2 people working on how we would realize our UI Mockup design and they both utilized different starting code. We tried to keep components from both designs, but it

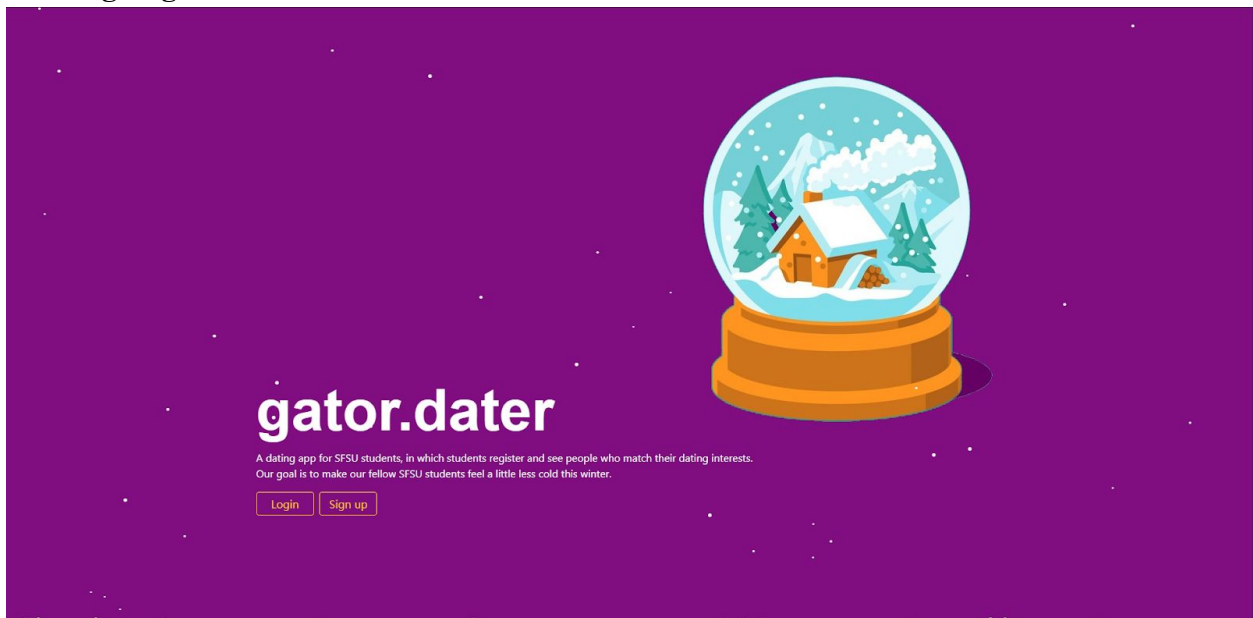
turned into a big merge conflict headache. So, we had to slowly scrap a lot of it, and then build up from there. A big problem in the beginning was that we had a lot done, but not all merged together. Merging broke a lot of working features when they seemingly had nothing to do with each other, and sometimes even just accepting all incoming changes made it not runnable.

Updating each other online on their progress was not cutting it, so we had to meet up a lot to touch bases and do some pair programming to merge together all our features and work on new ones as well.

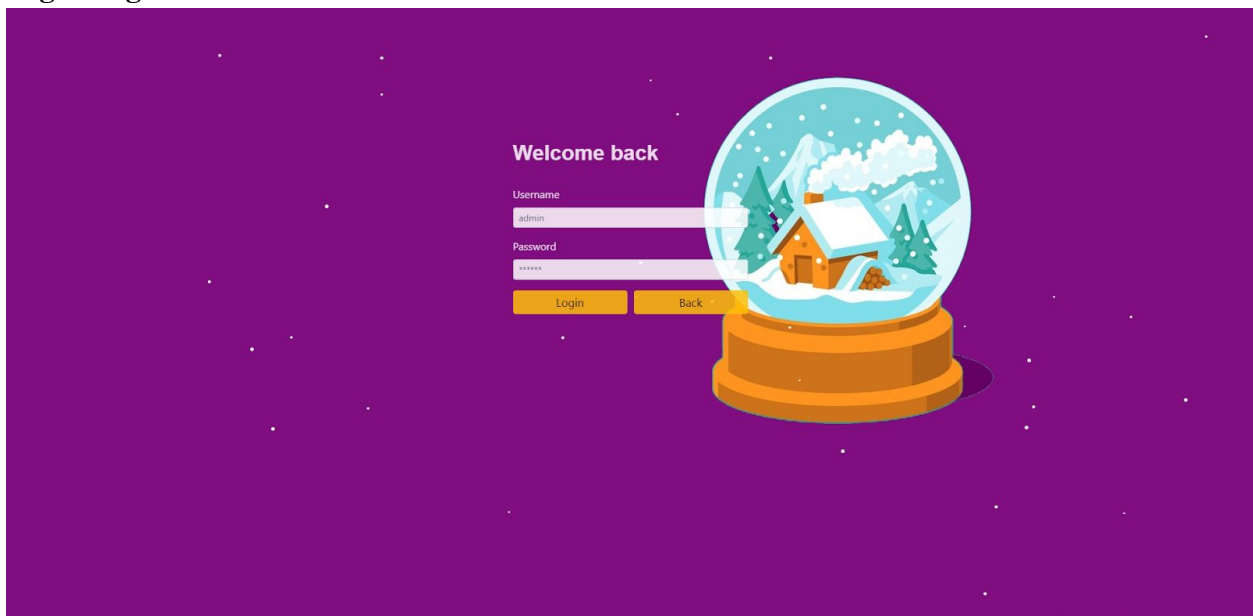
After finding a working rhythm regarding merging conflicts, we were able to have a working front end that had login and signup functions. The front end group was overly focused on making sure that the login and signup function worked well along with cookies. We made sure that after creating an user, the user will be immediately redirected to the profile page instead of logging in. In addition, we implemented the ability of the user to edit their profile information so that they can change their preferences and other details. A lot of time was allocated to that and therefore we had less time to implement our one on one messaging feature. So we had to scrap that idea and replace it with a “like” system. Once the like button is pressed, the username of that person is stored in the current user’s like array then the function will also check if the other user liked the current user’s profile, if it's a match then a modal will be displayed with each other email addresses. Lastly, we implemented a log out function that deleted the cookies and dispatch empty string on the input boxes to prevent an already filled out sign up page. This marked the end of our website functions and introduced the difficulty of implementing Docker and Kafka.

## Screenshots of All Screens:

### **Landing Page:**



### **Login Page:**



## Sign Up Page:

### Create Page

Username

username

Password

password

First Name

Brian

Last Name

Parra

Age


(18 - 65)

Email

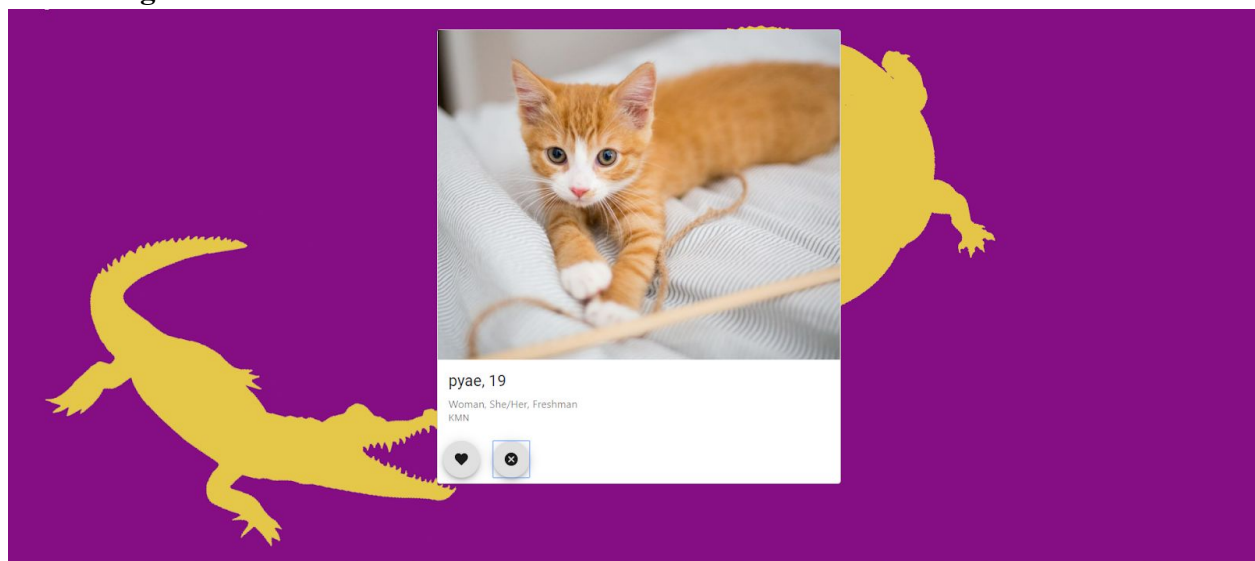
brianparra@mail.ohio.edu

Continue

Back



## Profile Page:



## Edit Profile Page/Sign Up (Page 2):

Update Profile:

Upload Files

Upload

First Name

Poo

Last Name

Plop

Password

—

Age

29

Email

poop@mail.sfsu.edu

College Year

(required)

I identify as..

—

What are your pronouns?

(required)

List me as..

(required)

Show me..

Men

Important details you'd like to share?

—

Complete